National University of Computer and Emerging Sciences



Lab Manual 9

CL461-Artificial Intelligence Lab

|  |  |
| --- | --- |
| Course Instructor | Dr. Mubashar Baig |
| Lab Instructor (s) | Nimra Abbas  Samia Akhter |
| Section | H |
| Semester | Spring 2022 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

Table of Contents

[1 Objectives 3](#_Toc72728147)

[2 Task Distribution 3](#_Toc72728148)

[3 Artificial Neural Network (ANN) 4](#_Toc72728149)

[3.1 Introduction – A Perceptron 4](#_Toc72728150)

[3.2 Multi-layer ANN Layers 4](#_Toc72728151)

[3.3 Back Propagation Algorithm 5](#_Toc72728152)

[3.4 Activation Functions 6](#_Toc72728153)

[3.5 ANN Use Cases 6](#_Toc72728154)

[4 Image Classification Using ANNs with Scikit-learn 6](#_Toc72728155)

[4.1 Labels 7](#_Toc72728156)

[4.2 Classification Evaluation Metrics 7](#_Toc72728157)

[5 Exercise (50 Marks) 7](#_Toc72728158)

[5.1 Load and Visualize the MNIST Dataset 8](#_Toc72728159)

[5.2 Test Your NN Classifier on Real-World Data 8](#_Toc72728160)

[6 Submission Instructions 8](#_Toc72728161)

# Objectives

After performing this lab, students shall be able tounderstandthe following:

* Artificial Neural Network (ANN)
* Image classification using ANN with scikit-learn

Pre-requisite: Lab manual 9 (Perceptron, Perceptron Training Rule etc)

# Task Distribution

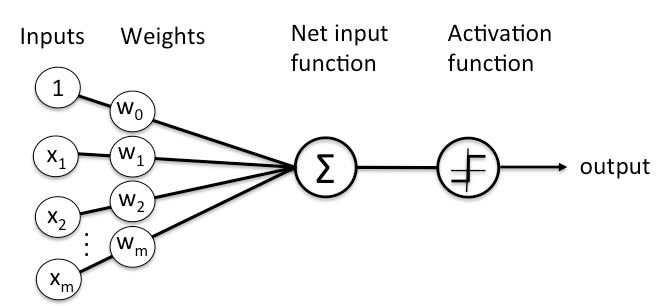
| **Total Time** | **170 Minutes** |
| --- | --- |
| Artificial Neural Network (ANN) | 15 Minutes |
| Image classification using ANNwith scikit-learn | 25 Minutes |
| Exercise | 120 Minutes |
| Online Submission | 10 Minutes |

# Artificial Neural Network (ANN)

The following section gives a brief introduction of perceptron, illustrates different components and layers of multilayer neural network along with backpropagation algorithm, and discusses different use cases of ANNs as well.

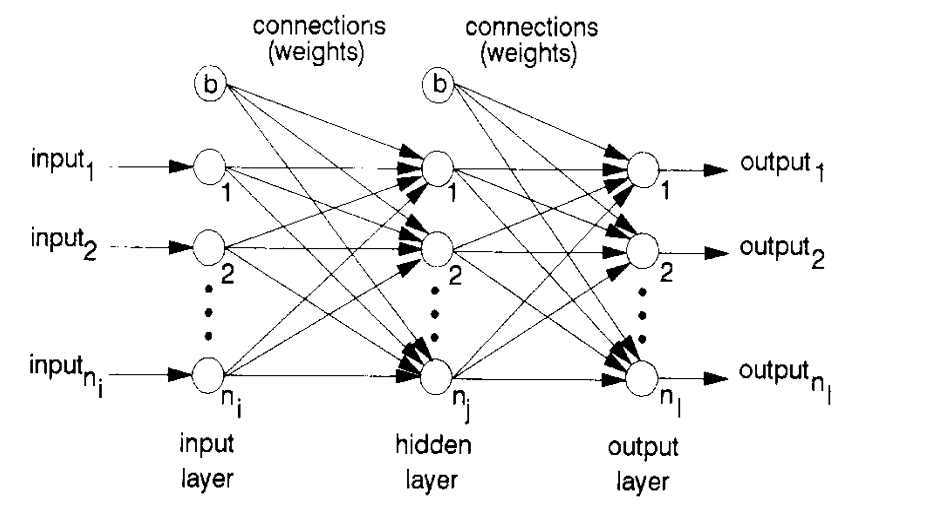
## Introduction – A Perceptron

An Artificial Neural Network is a computational network, based on biological neural networks that construct the structure of the human brain. Similar to how a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. A typical single neuron (also known as perceptron) is represented below.



## Multi-layer ANN Layers

A typical Neural Network contains a large number of artificial neurons called units arranged in a series of layers.ANNs have an input layer and output layer. Between these two layers there are other hidden layers that perform the mathematical computations that help determine the decision or action the machine should take. Ultimately, these hidden layers are in place to transform the input data into something the output unit can use.A typical multi-layer ANN is represented in the image below.



## Back Propagation Algorithm

It is the training or learning algorithm. It learns by example. If you submit to the algorithm the example of what you want the network to do, it changes the network’s weights so that it can produce desired output for a particular input on finishing the training.

The backprop algorithm cycles through two distinct passes, a forward pass followed by a backward pass through the layers of the network. The algorithm alternates between these passes several times as it scans the training data. Typically, the training data has to be scanned several times before the networks “learns” to make good classifications.

**Forward Pass:** Computation of outputs of all the neurons in the network.

The input is fed to the input layer, the neurons perform a linear transformation on this input using the weights and biases. Post that, an activation function is applied on this linear transformation to add non-linearity in the model in order to learn complex pattern from the data. Finally, the output from the activation function moves to the next hidden layer and the same process is repeated. This forward movement of information is known as the forward pass or propagation.



**Backward pass:** Propagation of error and adjustment of weights

The task is to make the output to the neural network as close to the actual (desired) output. Each of these neurons is contributing some error to the final output. How do you reduce the error? We try to minimize the value/weight of neurons that are contributing more to the error and this happens while traveling back to the neurons of the neural network and finding where the error lies. In order to minimize the error, the neural networks use a common algorithm known as “Gradient Descent”, which helps to optimize the task quickly and efficiently.

The one round of forwarding and backpropagation iteration is known as one training iteration aka “**Epoch**“.Most applications of feedforward networks and backprop require several epochs before errors are reasonably small.

## Activation Functions

Activation function is one of the building blocks of Neural Network. Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to **introduce non-linearity** into the output of a neuron.A neural network without an activation function is essentially just a linear regression model.

Popular types of activation functions are:

* Binary Step Function
* Linear Function
* Sigmoid
* Tanh
* Rectified Linear Unit (ReLU)
* Leaky ReLU
* ParameterisedReLU
* ELU
* Softmax

Refer to the following links and the lab session to understand activation functions in details:

* <https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/>
* <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>
* <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

## ANN Use Cases

ANNs are designed to spot patterns in the data. This makes ANNs an optimal solution for:

* classification (sorting data into predetermined categories)
* clustering (finding similar characteristics among data and pulling that data together into categories)
* making predictions from data (such as helping determine infection rates for COVID, the next catastrophic weather event or box-office smash).

Refer to the lab session for further details regarding ANNs.

# Image Classification Using ANNs with Scikit-learn

**Note:**

Find the complete scikit-learn user guide here: <https://scikit-learn.org/stable/user_guide.html>

Find the Multilayer Perceptron scikit-learn here: [https://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPClassifier.html#](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html%23)

Refer to the Colab Notebook (**CL461\_AI\_F\_Lab\_10\_Exercise.ipynb**) uploaded along this lab manual on Google Classroom.

Image classification is an amazing application of artificial intelligence. We can train a powerful algorithm to model a large image dataset. This model can then be used to classify a similar but unknown set of images.

We will build an Image classifier for the [Fashion-MNIST Dataset](https://github.com/zalandoresearch/fashion-mnist) (https://github.com/zalandoresearch/fashion-mnist). The Fashion-MNIST dataset is a collection of [Zalando's](https://research.zalando.com/) article images. It contains 60,000 images for the training set and 10,000 images for the test set data (*we will discuss the test and training datasets along with the validation dataset later*). These are 28 x 28 grayscale images and belong to the labels of 10 different classes.

## Labels

Each training and test example is assigned to one of the following labels:

| **Label** | **Description** |
| --- | --- |
| 0 | T-shirt/top |
| 1 | Trouser |
| 2 | Pullover |
| 3 | Dress |
| 4 | Coat |
| 5 | Sandal |
| 6 | Shirt |
| 7 | Sneaker |
| 8 | Bag |
| 9 | Ankle boot |

## Classification Evaluation Metrics

Evaluating a model is a major part of building an effective machine learning model. The most frequent classification evaluation metric that we use should be ‘**Accuracy**’. When building and optimizing your classification model, measuring how accurately it predicts your expected outcome is crucial. However, this metric alone is never the entire story, as it can still offer misleading results. You might believe that the model is good when the accuracy rate is 99%! However, it is not always true. This will be discussed later. For this lab, you will calculate the accuracy.

# Exercise (50 Marks)

Perform Image Classification on MNIST dataset. The MNIST database contains 60,000 training images and 10,000 testing images.These are small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.This is a similar problem as discussed above.

Some key steps to keep in mind while building this AI model are:

1. Explore MNIST handwritten digit classification dataset
2. Split the dataset into training, validation & testing data.
3. Develop a baseline model
4. Develop an improved model by tuning it
5. Finalize the model and make predictions on the test dataset

## Load and Visualize the MNIST Dataset

To get started, you can load and visualize the dataset using this code snippet in Python:

fromkeras.datasets import mnist

frommatplotlib import pyplot

# load dataset

(trainX, trainy), (testX, testy) = mnist.load\_data()

# summarize loaded dataset

print('Train: X=%s, y=%s' % (trainX.shape, trainy.shape))

print('Test: X=%s, y=%s' % (testX.shape, testy.shape))

# plot first few images

fori in range(9):

# define subplot

pyplot.subplot(330 + 1 + i)

# plot raw pixel data

pyplot.imshow(trainX[i], cmap=pyplot.get\_cmap('gray'))

# show the figure

pyplot.show()

## Test Your NN Classifier on Real-World Data

Once you have made your predictions using the test split of the MNIST dataset, you are required to test the accuracy of your model on real-world data.

In order to achieve this task, you are required to:

* Make your own handwritten digits from 0-9 using paint or any other online tool available.
* Use different brush strokes of different sizes to diversify your test data.
* Load these images as a dataset into your Python notebook.
* Convert these images into grayscale as well as resize them to 28 by 28 dimensions using Python (Use any Python library like OpenCV, PIL, or scikit-image).
* Provide these images to your trained digit classifier.
* Observe the model predictions.

For some inspiration, you can also visit this link: <https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html>

# Submission Instructions

Always read the submission instructions carefully.

* Rename your Jupyter notebook to your roll number and download the notebook as **.ipynb** extension.
* To download the required file, go to **File->Download .ipynb**
* Submit your 10 handwritten test images by adding them into a **zipped** folder.
* Also submit the **.ipynb** file. But **DO NOTzip** or **rar** your notebook.
* Submit the notebook and zipped folder containing images on Google Classroom under the relevant assignment.
* Late submissions will not be accepted.